The Parallel Performance of Schwarz Splitting

By Calvin J. Ribbens and Layne T. Watson

TR 90-34

The Parallel Performance of Schwarz Splitting

Calvin J. Ribbens*
Layne T. Watson*
Virginia Polytechnic Institute & State University

October 23, 1989

Abstract

We describe several experiments with implementations of Schwarz splitting algorithms for the numerical solution of elliptic partial differential equations. Results from experiments comparing several variations of the basic scheme in the context of real parallel implementations are presented. Both shared-memory and distributed-memory architectures are represented. Various possibilities for accelerating the Schwarz iteration and for solving the subdomain problems are considered, with particular emphasis on the resulting parallel performance.

1 Introduction

The term *Schwarz splitting* refers to a large family of algorithms for numerically solving partial differential equations (PDEs). The idea behind Schwarz splitting is very old, dating at least to 1869 when H. Schwarz proposed the *Schwarz alternating method* in the context of an existence proof for the Diriclet problem on the union of two overlapping regions (see Kantorovich and Krylov [10]). Miller [14] described a numerical analog to this procedure for approximately solving elliptic PDEs. More recently, Schwarz splitting has attracted considerable attention as a natural way to exploit parallelism in the numerical solution of PDEs. The thesis of Tang [20] was a significant contribution in this area. Papers by Lions [11,12] analyze Schwarz splitting from several different points of view. The convergence behavior of the Schwarz iteration is considered in Rodrigue and Simon [18,19], Rodrigue [17], Evans, et al. [4], and Goovaerts and Piessens [6]. Meier [13] and others have looked at improving the iteration by a SOR-like strategy.

Schwarz splitting is very closely related to the methods of *domain decomposition*. In fact, recently Chan [1] and Widlund [21] have shown how the two approaches are equivalent in certain cases. Two recent volumes (see Glowinski, et al. [5] and Chan, et al. [2]) contain many papers on domain decomposition. Of particular interest are papers by Gropp and Keyes [7] and by Haghoo and Proskurowski [9] which report on the parallel performance of some domain decomposition methods.

The basic idea of Schwarz splitting is to split the domain into overlapping subdomains, and to repeatedly solve the PDE on each of the subdomains independently (thus the potential for parallelism). Boundary conditions for subdomain boundaries which coincide with the original boundary are given by the original boundary conditions; boundary conditions for subdomain boundaries which lie inside a neighboring subdomain are typically derived from the approximate solution at

*Supported by DOE grant DE-FG05-88ER25068.

a previous iteration. The iteration continues until some stopping criterion is satisfied (typically the relative change in the solution is less than some tolerance). Rodrigue [17] characterizes this process as an "inner" and an "outer" iteration. The outer iteration is the Schwarz iteration over the subdomains. The inner iteration refers to the method used to solve the PDEs on the individual subdomains. It is not required that an iterative method be used to solve the subproblems, of course. We will simply refer to the "inner" and "outer" solution methods in this paper.

It is possible in certain simple cases to view Schwarz splitting algorithms strictly in terms of classic iterative methods for linear systems (see Hageman and Young [8], for example). In particular, the classic Schwarz alternating method can be viewed as a block Gauss-Seidel iteration on a certain matrix (called the *Schwarz Enhanced Matrix* by Tang [20]). In the same way, the simplest Schwarz splitting scheme, in which each inner solution is done simultaneously, using boundary conditions from neighboring subdomains at the previous iteration, can be viewed as a block Jacobi algorithm on the same system. This interpretation of Schwarz splitting is useful in that it allows the whole body of theory and practice from matrix iterative methods to be brought to this particular application. This view can also help to suggest methods for improving the performance of the algorithm. It is possible, for example, to define an analog of red-black SOR or of block Jacobi with conjugate gradient acceleration in the context of Schwarz splitting. The limitation of this purely linear-algebra view, however, is that it is not easily applied in all cases, especially when the problem is more general (e.g., variable coefficients, nonrectangular regions) or when methods more sophisticated than second-order finite differences are used to discretize the subdomain problems.

There are many variations to the basic Schwarz splitting scheme. The performance of the method is significantly affected by the amount of overlap between subdomains, the choice of stopping criteria for the Schwarz iteration, the method used to solve the individual subdomain problems, and the means by which new boundary conditions are derived. The primary purpose of this study is to experiment with several of these variations, to see what choices are best, especially when Schwarz splitting is viewed as a parallel algorithm. There are many ways that one could think about accelerating the convergence of the outer iteration, for example; but some of these approaches are not as attractive when considered in the light of communication costs, load balancing, and problem partitioning—issues that are only relevant in a parallel context.

2 Experiments

The experiments reported here are done using an implementation of Schwarz splitting built on the ELLPACK system (see Rice and Boisvert [15] and Cleveland and Ribbens [3]). ELLPACK is a powerful system for numerically solving elliptic PDEs. It includes a very high-level problem-statement language and a library of over fifty precompiled problem solving modules. This framework allows convenient implementation of Schwarz splitting, and it allows easy experimentation with the several variations of the algorithm we want to study.

All data reported is based on test problems (second-order, linear, self-adjoint elliptic problems, on two-dimensional rectangles, with known true solutions) given in Rice, et. al [16]. The experiments on a shared memory architecture are from a Sequent Symmetry S81 with 10 processors, running DYNIX. The distributed memory data is from an Intel iPSC hypercube with 32 processors, running XENIX. In each case, the fortran compiler available on the system is used, and double precision is used throughout.

In order to focus on a manageable number of factors, we hold constant several parameters in these studies. In particular, the overlap ratio between subdomains is set at 0.25. This means, for

example, that in the horizontal direction, one-fourth of a subdomain is overlapped on both the left and right. Each subdomain must be a rectangle, and we split in both the horizontal and vertical direction. We halt the Schwarz iteration when the maximum relative change in the solution is less than $(errest/100)$, where $errest$ is an estimate of the discretization error. The individual subdomain problems are discretized using ELLPACK module 5-POINT STAR (standard second-order finite differences).

Our primary focus is on the choice of solution method for the inner problems (given the choice of discretization), and on various approaches to accelerating the outer iteration. We are particularly interested in the answers to these questions in the context of actual parallel implementations. Further, we want to compare the performance of Schwarz splitting on shared-memory and distributed-memory machines. While there has been a considerable amount of work done on analyzing the theoretical properties of Schwarz splitting, there is still a lack of experience with real parallel implementations, and there is a lack of understanding of how differences in machine architectures affect the performance of this family of methods.

3 Results

In this section we summarize and report data from a series of experiments using our implementation of Schwarz splitting. We briefly evaluate this approach purely as a sequential method, and compare it with other sequential methods. We then compare three different schemes for the outer iteration, given a fixed choice for the inner iteration; and compare two choices for the inner iteration, given a fixed choice for the outer iteration. Parallel efficiency for several of these combinations is then reported, for both a shared-memory and a distributed-memory machine. Finally, we consider the problem of improving the accuracy of a solution, give a fixed number of processors. This last experiment is meant to ignore the traditional question: "which parallel algorithm is best as the number processors grows?"; instead we seek to answer the question: "how can I get a better answer, given the fact that I have a fixed number of processors?".

3.1 Comparison with other sequential methods

Table 1 contains data from a typical test problem. We report time (in seconds) and error for a given method on a given grid size. The error is estimated by taking the maximum relative error over a fixed 40×40 gird. Column "Grid" gives the number of grid lines in each direction. The four methods are briefly described in the caption of Table 1. They represent typical choices, commonly available in a software package such as ELLPACK. The data is for one processor on the Sequent Symmetry S81. Note that the order of the discretization (Spline Galerkin and Hermite Collocation are both fourth-order methods, while 5 Point Star is second-order) is of primary importance in the overall performance of the method. This same data is plotted in Figure 1, along with an additional curve indicating the performance on one processor of one of the Schwarz splitting methods described below. It is not surprising that the Schwarz splitting method is not competitive with the fourth-order methods, since it is based on a second-order discretization. However, we can see that as a sequential method, the Schwarz splitting approach still is not quite as efficient as the standard methods that are not based on splitting. Of course, it must be pointed out that there are still good possibilities for improvement in Schwarz splitting, and in fact this is one of the purposes of this research. Furthermore, it may be that Schwarz splitting is still attractive as a parallel method, since it parallelizes so easily and efficiently, while other methods may prove to be less attractive in

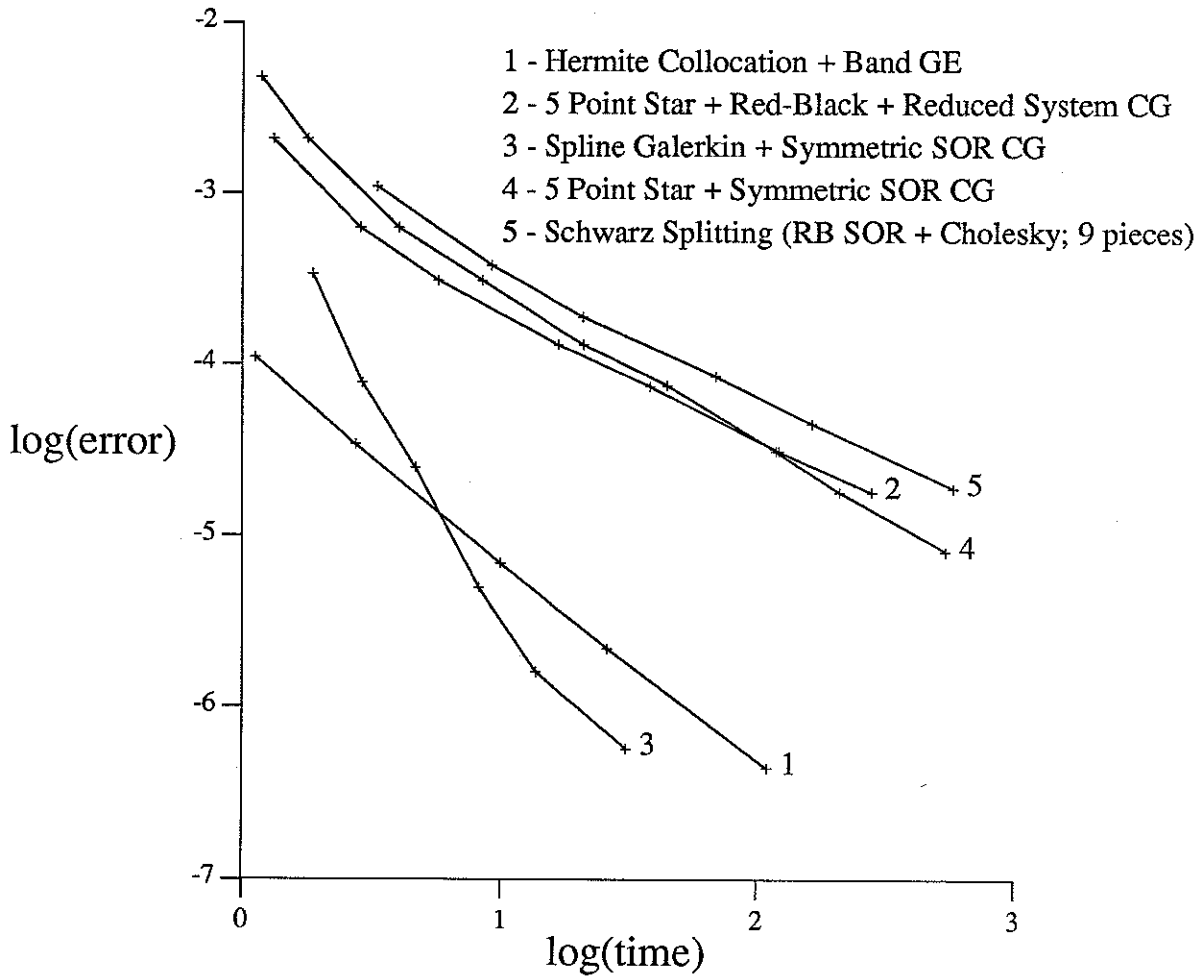

Figure 1: Log-log plot of error vs. time for four standard methods (no splitting) and for one Schwarz splitting method.

Table 1: Time and error for various grid sizes and various ELLPACK modules, with no splitting. Methods used are: (1) Hermite Collocation + Band GE, (2) 5 Point Star + Red-Black + Reduced System CG, (3) Spline Galerkin + Symmetric SOR CG, (4) 5 Point Star + Symmetric SOR CG.

	Method 1		Method 2		Method 3		Method 4	
Grid	Time	Error	Time	Error	Time	Error	Time	Error
5	0.37	5.7e-04	0.47	5.5e-02	1.85	3.4e-04		
7	1.12	1.1e-04	0.50	2.4e-02	2.87	7.8e-05		
9	2.70	3.4e-05	0.63	1.3e-02	4.62	2.5e-05	0.70	1.3e-02
13	9.93	6.9e-06	0.85	4.8e-03	8.17	5.0e-06	1.17	4.8e-03
17	26.25	2.2e-06	1.30	2.1e-03	13.68	1.6e-06	1.77	2.1e-03
25	109.72	4.4e-07	2.82	6.3e-04	31.05	5.7e-07	3.98	6.3e-04
33			5.68	3.1e-04			8.42	3.1e-04
49			16.87	1.3e-04			21.18	1.3e-04
65			38.40	7.4e-05			44.98	7.5e-05
97			121.73	3.1e-05			118.83	3.1e-05
129			278.38	1.8e-05			208.98	1.8e-05
193							542.65	8.1e-06

this regard.

3.2 Three choices for the outer iteration

As mentioned in Section 1 above, the classic Schwarz alternating method may be viewed as a Gauss-Seidel iteration. As such, it is not well-designed for a parallel implementation, since a typical inner solution cannot take place until at least one of its neighbors has completed the current iteration. The obvious parallel version of this is a Jacobi iteration in which all subdomain solves occur in parallel. The number of Schwarz iterations needed for convergence will be greater than for the Gauss-Seidel iteration, but the degree of parallelism is much higher. In the same spirit, we can derive a SOR-like version of Schwarz splitting. Again, the classic SOR iteration is not convenient for parallel implementation; but it is well known that a "red-black" ordering of the unknowns (in this case, the subdomains) allows half of the work to be done in parallel. In this section we compare the sequential performance of these two approaches, given that the inner solves are done using 5 Point Star and Linpack SPD Band (band Cholesky factorization). Note that the matrices corresponding to the inner problems do not change from iteration to iteration. The only changes that occur are in the right-hand side. Consequently, we can form and factor the linear systems once, and then simply update the right-hand side and forward- and back-solve to compute the new estimate of the solution at each iteration.

We also compare a third method for the outer iteration, based on a conjugate acceleration of the Jacobi iteration. This method is essentially the classic conjugate gradient algorithm applied to a slightly modified version of the discrete problem, with a block diagonal preconditioner. Algorithmically, this method is much more complex than the first two described in this section. In order to guarantee that the Schwarz Enhanced Matrix is symmetric and positive definite, we must modify the individual subproblems slightly (to "symmetrize" them) by introducing some extra coupling between subdomains.

Table 2: Schwarz iterations (SI), time, and error for various grid sizes and three choices for Schwarz iteration. Choices are: Jacobi, Red-Black SOR, and Jacobi CG. All cases solve the inner problem using Linpack SPD Band.

2×2 Subdomains

	Jacobi			Red-Black SOR			Jacobi CG		
Grid	SI	Time	Error	SI	Time	Error	SI	Time	Error
5	31	0.5	1.7e-02	11	0.2	1.7e-02	8	0.3	1.7e-02
9	36	2.6	3.2e-03	12	1.1	3.2e-03	12	1.5	3.2e-03
13	39	7.7	9.7e-04	13	3.3	9.7e-04	15	4.7	9.7e-04
17	41	16.9	4.4e-04	13	7.1	4.3e-04	17	10.7	4.3e-04
25	44	56.5	1.9e-04	14	23.9	1.8e-04	21	38.0	1.7e-04
33	46	130.7	1.0e-04	15	57.6	1.0e-04	24	91.9	9.9e-05
49	48	442.8	4.3e-05	15	197.6	4.2e-05	31	355.5	4.2e-05
65	50	1080.4	2.3e-05	16	507.7	2.3e-05	36	935.8	2.3e-05

3×3 Subdomains

	Jacobi			Red-Black SOR			Jacobi CG		
Grid	SI	Time	Error	SI	Time	Error	SI	Time	Error
5	59	1.8	7.4e-03	15	0.6	7.4e-03	12	0.7	7.4e-03
9	68	10.5	1.1e-03	17	3.3	1.1e-03	18	4.6	1.1e-03
13	74	31.3	3.9e-04	17	9.2	3.8e-04	23	14.6	3.6e-04
17	78	67.9	2.0e-04	19	21.1	1.9e-04	26	33.2	1.9e-04
25	83	223.1	8.9e-05	20	69.3	8.5e-05	33	120.1	8.5e-05
33	87	512.9	4.7e-05	21	162.6	4.5e-05	38	292.8	4.5e-05
49	93	1742.8	2.0e-05	23	580.3	1.9e-05	49	1134.1	1.9e-05

4×4 Subdomains

	Jacobi			Red-Black SOR			Jacobi CG		
Grid	SI	Time	Error	SI	Time	Error	SI	Time	Error
5	97	5.1	3.9e-03	19	1.3	3.9e-03	16	1.5	3.9e-03
9	113	30.3	5.3e-04	22	7.2	5.0e-04	25	10.4	5.0e-04
13	123	90.1	2.0e-04	25	22.2	1.8e-04	31	32.9	1.8e-04
17	129	193.4	1.2e-04	27	49.3	1.1e-04	37	78.0	1.1e-04
25	139	642.9	5.0e-05	30	168.2	4.6e-05	46	280.7	4.6e-05
33	145	1468.3	2.9e-05	32	396.8	2.7e-05	54	693.3	2.7e-05

Table 2 contains data from a typical case with these three methods. We report the error in each case, although it should be essentially the same for all three, since discretization error dominates. Column "grid" gives the number of grid lines in one direction on each subdomain. As expected, the number of Schwarz iterations is reduced substantially by SOR, when compared to Jacobi. The difference grows as the number of subdomains increases. The Jacobi CG method also takes fewer iterations than the Jacobi method, but its advantage is not as great as might be expected. It seems that this is primarily due to the increased coupling that is necessary in order to apply the CG procedure (i.e., we are really solving two different problems). Finally, it must be recalled that the red-black SOR scheme has only half of the potential parallelism of the other two methods. Consequently, in the context of a parallel machine with n processors, the most efficient subdomain decomposition for Jacobi or Jacobi CG might be into n subdomains, while for Red-Black SOR it might be more efficient to split into $2n$ subdomains. The relationship between the number of Schwarz subdomains and the number of processors is investigated further in Section 3.5 below.

3.3 Two choices for the inner iteration

The solution to an inner problem is essentially a solution to the "wrong" problem, since the boundary conditions are correct only in the limit. Thus it is worth considering whether an iterative method is more efficient for the inner problem than a direct method. With this strategy, one could do just a few iterations of the iterative method, enough to keep the overall Schwarz iteration converging, but at a cost substantially less than that of a direct method. Another common motivation for direct methods in the context of PDEs is memory constraints. This is especially relevant on a distributed memory machine such as the hypercube. Our Intel iPSC has only 0.5Mbyte of local memory on each processor, and much of that is needed for the operating system and for our own program code. With a direct method on the inner problems, and with each processor responsible for one subdomain, the finest grid that can be used is 25×25.

We consider using the SOR method to solve the inner subproblems. Rather than solving each subproblem to "complete" accuracy (i.e., to the level of discretization error), we simply do a fixed number of SOR iterations on each subproblem during each outer iteration. We refer to this method as "Incomplete SOR". It is necessary to iterate until "complete" accuracy is achieved during one last Schwarz iteration, after convergence of the outer iteration occurs. For the data reported in Table 3 we fix the number of SOR iterations at five. We find that the performance of the method is fairly insensitive to the number of SOR iterations done on the inner problem. In a similar context, Rodrigue [17] shows that only one iteration (of a Jacobi scheme) on the inner problem is enough to guarantee convergence of the outer Jacobi iteration. We can see that the number of outer iterations needed to converge is very much influenced by the number of SOR iterations on the inner problem; but the increase in time due to more outer iterations is roughly balanced by the savings due to cheaper iterations. Note that the number of Schwarz iterations for the Incomplete SOR method is very much greater than when the direct method is used on the inner problem. The total time (for one processor) is quite comparable, however. As the number of subdomains grows, the direct method seems to be best. However, it is likely that a more sophisticated iterative method, such as Symmetric SOR CG, would make using an iterative method for the inner problem more attractive.

3.4 Parallel efficiency of various Schwarz splitting methods

Tables 4 and 5 show typical parallel efficiencies for our Schwarz splitting implementations on the shared-memory and distributed-memory machine respectively. In Table 4 we give parallel efficiency

Table 3: Schwarz iterations (SI), time, and error for various grid sizes and two choices for solution method on subproblems. Choices are: Cholesky factorization and Incomplete SOR. All cases use Jacobi for the outer iteration.

2×2 Subdomains

	Cholesky			Incomplete SOR		
Grid	SI	Time	Error	SI	Time	Error
5	31	0.5	1.7e-02	32	0.6	1.7e-02
9	36	2.6	3.2e-03	38	2.7	3.2e-03
13	39	7.7	9.7e-04	45	7.4	9.7e-04
17	41	16.9	4.4e-04	51	15.5	4.4e-04
25	44	56.5	1.9e-04	67	47.9	1.9e-04
33	46	130.7	1.0e-04	84	115.8	1.0e-04
49	48	442.8	4.3e-05	116	384.5	4.4e-05
65	50	1080.4	2.3e-05	149	914.7	2.4e-05

3×3 Subdomains

	Cholesky			Incomplete SOR		
Grid	SI	Time	Error	SI	Time	Error
5	59	1.8	7.4e-03	60	2.3	7.4e-03
9	68	10.5	1.1e-03	75	11.4	1.1e-03
13	74	31.3	3.9e-04	90	31.6	3.9e-04
17	78	67.9	2.0e-04	109	71.7	2.0e-04
25	83	223.1	8.9e-05	148	236.0	8.9e-05
33	87	512.9	4.7e-05	183	545.4	4.8e-05
49	93	1742.8	2.0e-05	260	1908.2	2.0e-05

4×4 Subdomains

	Cholesky			Incomplete SOR		
Grid	SI	Time	Error	SI	Time	Error
5	97	5.1	3.9e-03	98	6.5	3.9e-03
9	113	30.3	5.3e-04	125	32.9	5.3e-04
13	123	90.1	2.0e-04	154	95.4	2.0e-04
17	129	193.4	1.2e-04	185	209.0	1.2e-04
25	139	642.9	5.0e-05	258	715.1	5.0e-05
33	145	1468.3	2.9e-05	321	1709.8	2.9e-05

Table 4: Parallel efficiency for various grid sizes and three choices for Schwarz iteration. Choices are: Jacobi, Red-Black SOR, and Jacobi CG. All cases solve the inner problem using Linpack SPD Band (Cholesky factorization). Column Ei is parallel efficiency with i processors.

2×2 Subdomains

	Jacobi		RB SOR	Jacobi CG	
Grid	E2	E4	E2	E2	E4
5	0.95	0.82	0.90	0.74	0.78
9	0.97	0.93	1.00	0.89	0.81
13	0.99	0.97	0.99	1.00	0.92
17	0.98	0.98	1.00	0.99	0.94
25	0.99	0.98	1.00	1.00	0.98
33	1.00	0.98	0.99	1.00	0.99
49	0.99	0.99	1.00	1.00	0.98
65	1.00	0.99	0.99	1.00	0.99

3×3 Subdomains

	Jacobi		RB SOR		Jacobi CG	
Grid	E3	E9	E3	E5	E3	E9
5	0.84	0.68	0.85	0.71	0.75	0.32
9	0.88	0.74	0.88	0.76	0.93	0.69
13	0.89	0.78	0.88	0.77	0.98	0.85
17	0.92	0.81	0.88	0.81	0.98	0.91
25	0.94	0.87	0.89	0.83	0.98	0.94
33	0.96	0.82	0.89	0.86	0.99	0.93
49	0.98	0.92	0.89	0.87	0.99	0.95

4×4 Subdomains

	Jacobi			RB SOR			Jacobi CG		
Grid	E2	E4	E8	E2	E4	E8	E2	E4	E8
5	0.94	0.85	0.77	1.00	0.91	0.81	0.92	0.77	0.53
9	0.98	0.88	0.85	1.00	0.87	0.81	0.97	0.92	0.81
13	0.99	0.91	0.89	0.99	0.91	0.83	0.99	0.97	0.90
17	0.99	0.93	0.88	0.99	0.93	0.86	0.99	0.97	0.93
25	1.00	0.95	0.92	0.99	0.95	0.89	0.99	0.98	0.95
33	1.00	0.96	0.94	0.99	0.96	0.91	0.99	0.98	0.95

Table 5: Parallel efficiency for various grid sizes and cube dimensions. In all cases the method used is Jacobi outer iteration and Incomplete SOR inner iteration. Column Ei is efficiency with 2^i processors.

Grid	E1	E2	E3	E4	E5
5	0.98	0.89	0.62	0.26	0.07
9	0.99	0.98	0.94	0.82	0.29
13	0.99	0.98	0.97	0.93	0.69

on the Sequent Symmetry for the three methods considered in Section 3.2 on a typical test problem. Table 5 gives results for the same problem on the hypercube, using Jacobi outer iteration and Incomplete SOR inner iteration. In general, we see that parallel efficiency is very good, especially for the shared-memory data, and for fine grids. This is not surprising, since Schwarz splitting is a method whose most obvious property is the ease with which it can be parallelized. It would be disappointing indeed, if these algorithms did not achieve nearly linear speedup on shared-memory machines with a modest number of processors. Note that the number of processors is chosen so that the work-load (number of subdomains) is as evenly distributed as possible. It is impossible to achieve completely uniform load-balancing for Red-Black SOR in the 3×3 subdomain case, for example. Of course, the load-balancing problem would be mitigated if the domain was split into very many more subdomains than there are processors. In Section 3.5 below we consider whether this is a good idea.

The distributed memory performance is less impressive, although the efficiency for the largest problem is not too bad (69%). It must be pointed out that the problem sizes reported here are very much constrained by what one can fit on one processor. We see again a common situation in evaluating parallel algorithms for distributed memory machines—fixed problem-size speedups are not impressive and there is difficulty in finding problem sizes that both fit on one processor and are efficiently solved using many processors. Of course, scaled problem-size speedup would be a much more impressive measure in this context, though some would argue it is unfairly optimistic or even misleading.

3.5 Improving a solution

As briefly indicated above, the purpose of this last experiment is to consider a situation in which the number of processors is fixed, but a more accurate solution is required. Within the Schwarz splitting framework there are two obvious alternatives: refine the grid in each subdomain, or increase the number of subdomains while holding the grid size per subdomain constant. As long as Schwarz splitting is a less than optimal sequential method, it is clear that the former choice is the best one—it is better to solve the same number of subproblems with a finer grid than to solve more subproblems of the same size. Tables 6 and 7 contain data for typical cases for the shared-memory and distributed-memory machines, respectively. The base case in Table 6 has 9 subdomains and a 5×5 grid on each subdomain. The base case in Table 7 has 16 subdomains and a 5×5 grid on each subdomain. Figures 2 and 3 illustrate even more clearly the conclusion. In each case, the curve labeled "4" represents the decision to refine the grid on each subdomain. The other 3 curves represent the decision to add subdomains (given a fixed grid per subdomain of 5×5, 9×9, and 13×13, respectively).

Table 6: Schwarz iterations (SI), time, and error for various combinations of number of subdomains and grid size per subdomain. All cases use Jacobi outer iteration, Incomplete SOR inner iteration, and 9 processors on the Sequent Symmetry.

Pieces	Grid	SI	Time	Error
3×3	5	60	0.4	7.4e-03
3×3	9	75	1.5	1.1e-03
3×3	13	90	3.7	3.9e-04
3×3	17	109	8.7	2.0e-04
3×3	25	148	28.9	8.9e-05
3×3	33	183	67.2	4.8e-05
3×3	49	260	252.8	2.0e-05
3×3	65	338	645.4	1.2e-05
6×6	5	204	4.1	1.3e-03
6×6	9	265	18.4	2.6e-04
6×6	13	327	51.5	1.1e-04
9×9	5	436	18.5	5.9e-04
9×9	9	573	88.5	1.4e-04
9×9	13	710	255.9	6.1e-05
12×12	5	757	55.7	3.9e-04

Table 7: Schwarz iterations (SI), time, and error for various combinations of number of subdomains and grid size per subdomain. All cases use Jacobi outer iteration, Incomplete SOR inner iteration, and a cube of dimension 4 (i.e., 16 processors).

Pieces	Grid	SI	Time	Error
4×4	5	96	24.2	3.9e-03
4×4	9	124	28.3	5.7e-04
4×4	13	152	53.2	2.3e-04
4×4	17	184	110.9	1.2e-04
4×4	25	257	357.9	5.3e-05
4×4	33	320	856.7	3.0e-05
8×8	5	347	89.3	7.0e-04
8×8	9	456	235.0	1.6e-04
8×8	13	566	685.0	7.0e-05
12×12	5	755	232.4	4.0e-04
12×12	9	995	1102.7	9.7e-05

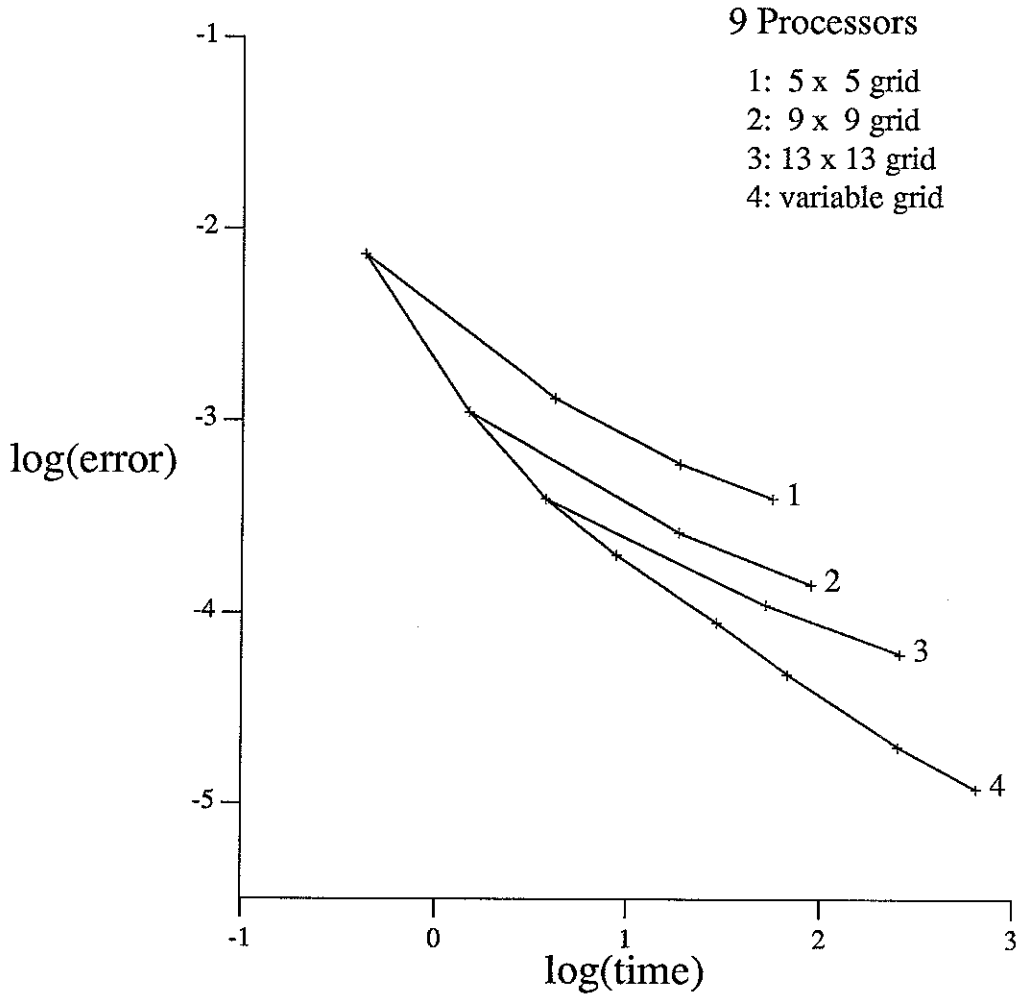

Figure 2: Log-log plot of error vs. time for four different approaches to improving a solution, given a fixed number of processors. Data is for the Sequent.

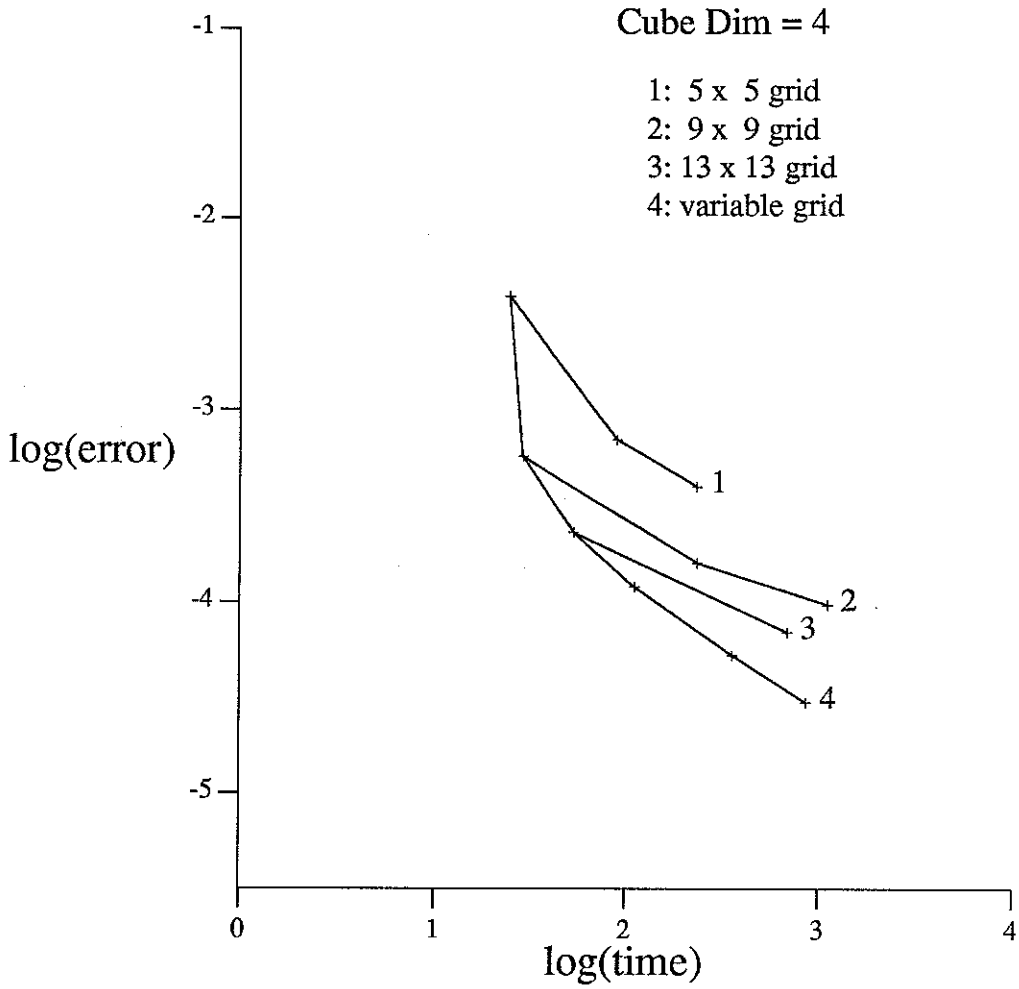

Figure 3: Log-log plot of error vs. time for four different approaches to improving a solution, given a fixed number of processors. Data is for the Hypercube.

4 Conclusions and Future Work

It is clear that Schwarz splitting algorithms are easy to parallelize, especially on shared-memory machines. As a sequential method, there still needs to be improvement however, before Schwarz splitting can be applied efficiently. There are many possibilities for such improvement. With only a few improvements suggested by analogs in classical matrix iterative analysis, we see dramatic improvement in the performance of the method. It is likely that further progress in this direction is possible. We are currently investigating preconditioned conjugate gradient schemes that do not require the modification to the problem mentioned in Section 3.2. We are also considering polynomial acceleration methods in this context. The inner problem solving method can also be improved. A better choice of iterative method than SOR is likely to improve the "incomplete" inner solve approach, for example. The use of different types of boundary conditions on the interior boundaries has also drawn much interest of late.

It is also clear that the number of subdomains into which the domain is divided should not grow arbitrarily. The overall convergence of the method suffers when very many subdomains are defined, each with a very coarse grid. For massive parallelism (e.g., thousands of processors), is would seem that one subdomain for each processor is the wrong approach. Instead, a multi-level approach is suggested, in which groups of processors cooperate to solve the inner problem on one subdomain. At the highest level the domain should be decomposed into a fairly small number of subdomains. The convergence of the overall scheme is dictated in large part by this decomposition at the highest level. Within a subdomain however, it is possible that further decompositions could be done. The parallels with multigrid strategies for PDEs are obvious. These connections should be pursued actively; a few researchers are beginning to do so in connection with (non-overlapping) domain decomposition. It is worth considering whether a "recursive Schwarz splitting" strategy might be effective, in which the subdomain problems are themselves solved using Schwarz splitting. We intend to pursue these ideas in the near future.

Finally, as has been mentioned, the relationship between non-overlapping and overlapping (Schwarz) domain decomposition is now much more clearly understood. The unification of these two approaches should bring benefits of both a theoretical and practical nature. The domain decomposition literature is dominated by work on preconditioners for conjugate gradient methods. Since one variation of Schwarz splitting can be viewed as domain decomposition with a certain preconditioner (see Chan [1]) it seems likely that other variations of Schwarz splitting will be suggested by other proposed preconditioners.

References

[1] T. F. Chan and D. Goovaerts, "Schwarz=Schur: Overlapping versus nonoverlapping domain decomposition", Dept. of Mathematics Report CAM 88-21, UCLA, 1988.

[2] T. Chan, R. Glowinski, J. Periaux and O. Widlund, eds., *Domain Decomposition Methods*, SIAM, Philadelphia, 1989.

[3] B. W. Cleveland and C. J. Ribbens, "Schwarz splitting using ELLPACK", Dept. of Computer Science Report 88-2, Virginia Polytechnic Institute and State University, 1988

[4] D. J. Evans, Shao Jianping and Kang Lishan, "The convergence factor of the parallel Schwarz overrelaxation method for linear systems", *Parallel Comput.* 6(1988), pp 313–324.

[5] R. Glowinski, G. Golub, G. Meurant and J. Periaux, eds., *Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988.

[6] D. Goovaerts and R. Piessens, "The rate of convergence of Schwarz iteration in domain decomposition methods for separable equations on a rectangle", Dept. of Computer Science Report TW 90, Katholieke Universiteit Leuven, Belgium, 1987.

[7] W. D. Gropp and D. E. Keyes, "Domain decomposition on parallel computers", in *Domain Decomposition Methods*, (T. Chan, R. Glowinski, J. Periaux and O. Widlund, eds.), SIAM, Philadelphia, 1989, pp 260–268.

[8] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.

[9] M. Haghoo and W. Proskurowski, "Parallel efficiency of a domain decomposition method", in *Domain Decomposition Methods*, (T. Chan, R. Glowinski, J. Periaux and O. Widlund, eds.), SIAM, Philadelphia, 1989, pp 269–281.

[10] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*, Interscience, New York, 1964.

[11] P. L. Lions, "On the Schwarz alternating method I", in *Domain Decomposition Methods for Partial Differential Equations*, (R. Glowinski, G. Golub, G. Meurant and J. Periaux, eds.), SIAM, Philadelphia, 1988, pp 1–42.

[12] P. L. Lions, "On the Schwarz alternating method II: stochastic interpretation and order properties", in *Domain Decomposition Methods*, (T. Chan, R. Glowinski, J. Periaux and O. Widlund, eds.), SIAM, Philadelphia, 1989, pp 47–70.

[13] U. Meier, "Two parallel SOR variants of the Schwarz alternating procedure", *Parallel Comput.*, 3(1986), pp 205–215.

[14] K. Miller, "Numerical analogs to the Schwarz alternating procedure", *Numer. Math.*, 7(1965), pp 91–103.

[15] J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[16] J. R. Rice, E. N. Houstis and W. R. Dyksen, "A population of linear, second order elliptic partial differential equations on rectangular domains", *Math. Comp.*, 36(1981), pp 479–484.

[17] G. Rodrigue, "Inner/outer iterative methods and numerical Schwarz algorithms", *Parallel Comput.*, 2(1985), pp 205–218.

[18] G. Rodrigue and J. Simon, "Jacobi splittings and the method of overlapping domains for solving elliptic pdes", in *Advances in Computer Methods for Partial Differential Equations V*, R. Vichnevetsky and R.S. Stepleman, eds., IMACS, Brussels, 1984, pp 383-386.

[19] G. Rodrigue and J. Simon, "A generalization of the numerical Schwarz algorithm", in *Computing Methods in Applied Sciences and Engineering VI*, R. Glowinski and J.L. Lions, eds., North-Holland, Amsterdam, 1984, pp 273-283.

[20] W. P. Tang, "Schwarz splitting and template operators", Ph.D. thesis, Stanford University, 1987.

[21] O. Widlund, "Towards a unified theory for domain decomposition methods for elliptic problems", presentation, SIAM Conference on Domain Decomposition Methods, Houston, March 1989.